National Défense
Defence nationale

UNCLASSIFIED

DREV REPORT 4319/84
FILE: 3632D-001
JANUARY 1984

CRDV RAPPORT 4319/84
DOSSIER: 3632D-001
JANVIER 1984

A FORTRAN CODING OF THE SPECTRAL TEST FOR ASSESSING THE RANDOMNESS AND THE MERIT OF LINEAR CONGRUENTIAL GENERATORS

R. Carling

K. Levine

Centre de Recherches pour la Defense
Defence Research Establishment
Valcartier Quebec

BUREAU - RECHERCHE ET DÉVELOPPEMENT
MINISTÈRE DE LA DÉFENSE NATIONALE
CANADA

Canada

NON CLASSIFIÉ

RESEARCH AND DEVELOPMENT BRANCH
DEPARTMENT OF NATIONAL DEFENCE
CANADA

DREV R-4319/84 UNCLASSIFIED CRDV R-4319/84
FILE: 3632D-001 DOSSIER: 3632D-001

A FORTRAN CODING OF THE SPECTRAL TEST FOR ASSESSING THE RANDOMNESS AND THE MERIT OF LINEAR CONGRUENTIAL GENERATORS

by

R. Carling and K. Levine

CENTRE DE RECHERCHES POUR LA DÉFENSE

DEFENCE RESEARCH ESTABLISHMENT

VALCARTIER

Tel: (418) 844-4271

Québec, Canada January/janvier 1984

NON CLASSIFIÉ

ABSTRACT

In this report, a mathematical algorithm known as the spectral test for assessing the randomness and merit of linear congruential generators is presented. Details of how to use our FORTRAN coding of the spectral test on the Honeywell 560 are given. Widely used generators such as RANDU (NUMGEN) and two IMSL generators are assessed by it. Alternative generators are proposed for bad ones not passing the spectral test.

RÉSUMÉ

Dans ce rapport on présente un algorithme mathématique, connu sous le nom d'essai spectral, comme moyen d'évaluer la répartition et le mérite des générateurs congruents linéaires. On explique le mode d'emploi de notre codage en FORTRAN de l'essai spectral pour ordinateur Honeywell 560. On évalue le mérite de générateurs souvent employés en simulation tels que RANDU (NUMGEN) et de deux générateurs de IMSL. On propose d'autres générateurs pour remplacer ceux qui échouent l'essai spectral.

TABLE OF CONTENTS

ABSTRACT/RÉSUMÉ 1

1.0 INTRODUCTION 1

2.0 HISTORY AND BACKGROUND 2

3.0 MATHEMATICAL PRELIMINARIES 5

4.0 THE SPECTRAL TEST COMPUTER PROGRAM 11

5.0 HOW TO USE THE SPECTRAL TEST COMPUTER PROGRAM 14

6.0 ASSESSMENT OF COMMONLY USED GENERATORS AND PROPOSAL OF ALTERNATIVES 18

7.0 NORMAL DEVIATES 22

8.0 CONCLUSION 23

9.0 ACKNOWLEDGEMENTS 24

10.0 REFERENCES 25

TABLES I and II

FIGURES 1 to 4

APPENDIX A: Listing of Three Generators of Uniform Deviates and One Generator of Normal Deviates for the Honeywell 560 26

1.0 INTRODUCTION

A discrete event Monte Carlo simulation can be thought of as a statistical experiment. In such a simulation, there are various units that can exist in a different number of states. As time progresses in the simulation, the state of units changes giving rise to a sequence of events. The sequence of events produced depends on fixed and random factors. The latter are simulated by random sampling from normal and uniform probability distributions. If the sampling is not truly random, statistics measuring the efficiency, performance and other important characteristics of the simulation could be biased. To carry out our systems studies by computer simulations, we make extensive use of random number generators. In these simulations, random numbers are often required in groups of two, three, four and even five. It is then important to ascertain that the random number generator can reliably produce such groups of random numbers.

After looking through the literature, we singled out the spectral test as being an excellent criterion (Ref. 1) of assessing the randomness of our random number generators. The object of this work was therefore to program the spectral test, and use it to assess our random number generator as well as other widely used ones.

We have developed a multiple precision version of the spectral test described in Ref. 1, pp 89-113, because of the difficulty of representing large integers on a 32-bit computer. We have used this program to assess RANDU (or NUMGEN as it is sometimes called) as well as the two IMSL subroutines, GGUBS and GGUBT. We have also used the spectral test program as a probe to find other good random number generators. Finally, we developed a subroutine for calculating normal deviates from two of the good random number generators found by our spectral test program.

This work was performed at DREV between May and August, 1982 under PCN 32D01, Tactical ADP Technology - General.

2.0 HISTORY AND BACKGROUND

A large class of Monte Carlo simulations depend on successive draws of uniform deviates from the unit interval (0, 1) and on normal deviates drawn from the real line. One way of generating uniform deviates between 0 and 1 is to use a linear congruential generator of the form

$$x_n = (ax_{n-1} + c) \bmod m \qquad [1]$$

where a, c, m are integers. The pseudorandom number x_n generated at the n^{th} step is then divided by m to obtain a real number between 0 and 1. The idea of the linear congruential generator [1] was originally stated by D.H. Lehmer in 1949 for the case c = 0 and extended to cases with arbitrary values of c by Rotenberg in 1960 (see Ref. 1, pp 9-10). The sequence $x_0, x_1, x_2, \ldots$ generated by [1] will eventually repeat. The number of distinct integers in the sequence is called the period and denoted by $\lambda(a, c, m, x_0)$. It turns out that in general, linear congruential generators with nonzero c have longer periods than generators with c = 0. It turns out also that the randomness of [1] tends to increase with greater periods. Hence it is useful to consider sequences with nonzero c. In 1967, MacPherson and Coveyou (Ref. 2) considered the linear congruential generator [1] to be generating t dimensional "waves". The numbers $Z_1^2 + \ldots + Z_t^2$ such that $Z_1 + \ldots + a^{t-1} Z_t \equiv 0 \bmod m$ in their treatment were the wave "frequencies" or points in the "spectrum" defined by the random number generator, hence the name spectral test. In this test, low-frequency

waves were the most damaging to randomness. Other authors have pointed out that the spectral test could be understood in terms of distances between hyperplanes containing successive t-tuples $(x_1, x_2, \ldots, x_t)$ coming from [1]. In this context, the quantity $x_1^2 + x_2^2 + \ldots + x_t^2$ represents the reciprocal of the distance between hyperplanes. Dieter and Gosper (see Ref. 1, pp 94-98) have developed an efficient computational algorithm for calculating these distances, by formulating the distance finding problem as a quadratic programming problem. The latter is solved iteratively. A Euclidean algorithm finds the distance between lines in two-space, then a process equivalent to Gramm-Schmidt orthogonalization calculates the distance between planes in three-space from the result obtained in two dimensions. The iteration ends when the distance between hyperplanes in six-space is calculated. It turns out that small values of the distance between hyperplanes is indicative of a good random number generator. The reason for this will be explained in the next chapter.

Marsaglia has pointed out an inherent weakness of linear congruential generators of the form [1]. In his paper (Ref.3) he proves that points in n-space produced by the congruential generator [1] must fall on a lattice with a unit-cell volume at least m^{n-1}. There is no restriction on a or c; this means that all congruential random number generators must be considered unsatisfactory in terms of lattices containing the points they produce, for a good generator of random integers should have an n-lattice with unit-cell volume 1. (The n-lattice of a random number generator is defined as follows: if the generator produces integers $r_1, r_2, r_3, \ldots$, let $w_1 = (r_1, r_2, \ldots, r_n)$, $w_2 = (r_2, r_3, \ldots, r_{n+1})$, ... be the set of possible points in n space formed from n successive r's. The n-lattice of the generator is the set of all integral linear combinations of points from this set translated to include the origin, i.e. all integral linear combinations

of the points $w_2 - w_1$, $w_3 - w_1$, ... The unit-cell volume of the n-lattice is the greatest common divisor of the volumes of parallelepipeds formed from any n + 1 points of the lattice.) Very good congruential generators that pass the spectral test will of course suffer from this weakness. Marsaglia (Ref. 4) suggests that random numbers could be generated by the aid of three linear congruential generators. The method used is a composite generator which relies on three basic congruential generators

$$\begin{aligned} l_n &= 65539\ l_{n-1} \bmod 2^{32} \\ m_n &= 33554433\ m_{n-1} \bmod 2^{32} \\ k_n &= 362436069\ k_{n-1} \bmod 2^{32} \end{aligned} \qquad [2]$$

One starts with 128 initial odd integer values stored in the locations N(1), N(2), ..., N(128), and uses sequences l_n and m_n to choose one out of these 128 locations. The random number generated is based on the number in that location, say N(J), as well as l_n and m_n. The location N(J) is then refilled with a number from generator k_n. Thus, except for the initial 128 values, the vector N(1), ..., N(128) contains numbers from the sequence k_n. This method of producing random numbers could not be assessed by the spectral test.

3.0 MATHEMATICAL PRELIMINARIES

A linear congruential sequence is a sequence of integers $\{x_0, x_1, \ldots, x_n, x_{n+1}, \ldots\}$ defined by

$$x_{n+1} = (ax_n + c) \bmod m \qquad [3]$$

where a, c, m are integers and are respectively called multiplier, constant and modulus. x_0 is called the seed.

For example, the sequence defined by

$$x_{n+1} = (6x_n + 7) \bmod 10, \; x_0 = 1 \qquad [4]$$

generates $\{1, 3, 5, 7, 9, 1, 3, 5, 7, 9, \ldots\}$. We notice that the sequence contains only five distinct elements and begins to cycle again after 9. The number of distinct elements generated by the linear congruential sequence [3] is called the period. For eq. 4 the period is 5.

We shall now state without proof three useful theorems enabling the reader to calculate the period of an arbitrary linear congruential generator by inspection. The proofs of these theorems are to be found in Ref. 1, pp 16-20.

Theorem 1-1 The period of the linear congruential generator [3] is m if and only if:

a) c is relatively prime to m, i.e. they have no common prime divisors;
and
b) if p divides m, then p divides a-1 (p is a prime number);
or
c) if 4 divides m, then 4 divides a-1.

We need another definition before being able to state the next two theorems. Define the function λ on the positive integers by

$$\lambda(2) = 1,\ \lambda(4) = 2,\ \lambda(2^e) = 2^{e-2},\ e \geq 3$$
$$\lambda(p^e) = p^{e-1}(p-1),\ p \text{ prime and} \neq 2 \qquad [5]$$
$$\lambda(p_1^{e_1}p_2^{e_2}\ldots p_n^{e_n}) = \text{lcm}\left(\lambda(p_1^{e_1}),\ \lambda(p_2^{e_2}),\ \ldots,\ \lambda(p_n^{e_n})\right)$$

lcm meaning the least common multiple. We say that a positive integer a is a primitive element modulo m if $a^{\lambda(m)} \equiv 1 \bmod (m)$ and $a^{\lambda} \not\equiv 1 \bmod (m)$ for any λ smaller than $\lambda(m)$. Thus if $m = 4$, $\lambda(4) = 2$ according to [5] and we note that $3^2 = 9 \equiv 1 \bmod (4)$ and $3 \not\equiv 1 \bmod (4)$. This means that 3 is a primitive element mod (4). It is clear that 1 and 2 are not primitive elements mod (4).

We now state the theorem on periods of linear congruential sequences with the constant term equal to zero.

Theorem 1-2 The period of a linear congruential sequence with the constant term equal to zero is $\lambda(m)$ if and only if:

a) x_0 is relatively prime to m, and
b) a is a primitive element modulo m.

The definition of a primitive element does not allow us to characterize it easily. Hence we have the following:

Theorem 1-3 a is a primitive element modulo p^e if and only if:

a) $p^e = 2$, a is odd; $p^e = 4$, $a \equiv 3 \bmod 4$; $p^e = 8$, $a \equiv 3$ or 5 or 7 mod 8; $p = 2$, $e \geq 4$, $a \equiv 3$ or 5 mod 8;

or

b) p is odd, e = 1, $a \not\equiv 0 \bmod p$ and $a^{(p-1)/q} \not\equiv 1 \bmod p$ for any prime divisor q of p - 1;

or

c) p is odd, e > 1, a satisfies b) and $a^{p-1} \not\equiv 1 \pmod{p^2}$

To illustrate the use of these theorems consider the random number generator NUMGEN which is represented by the FORTRAN statements

```
      SUBROUTINE NUMGEN (IAL, UNIF)
      DOUBLE PRECISION MODLESS1
      DATA MODLESS1/2147483647/
      IAL = IAL * 65539
      IF (IAL.GE.0) GO TO 10
      IAL = IAL + 2147483647 + 1
   10 UNIF = IAL/MODLESS1
      RETURN
      END
```

The multiplier of NUMGEN(RANDU) is $65539 = 2^{16} + 3$. Using condition a) of Theorem 1-3, it is clearly a primitive element mod 2^{31}. The input integer IAL to the subroutine NUMGEN is always a large odd integer and, hence, is relatively prime to 2^{31}. Moreover, if the result of the fixed-point multiplication by 65539 exceeds $2^{31} - 1$, the computer (Honeywell 560) subtracts multiples of 2^{32} until the result is an integer lying between -2^{31} and $2^{31} - 1$. If this result is negative, our computer program adds 2^{31} to the negative number thus obtaining a nonnegative number lying between 0 and $2^{31} - 1$. Thus, the FORTRAN coding of NUMGEN given above represents the linear congruential sequence

$$x_n = (2^{16} + 3)\, x_{n-1} \bmod 2^{31} \qquad [6]$$

By Theorem 1-2, the period of the congruential sequence [6] is 2^{29}.

In order to explain the spectral test, we shall now limit ourselves to two types of linear congruential generators, i.e. the first for which $c \neq 0$ and the period is equal to the modulus and the second kind for which $c = 0$ and m is prime. In the second case the period is $m - 1$, but for purposes of the spectral test when m is large, one can take the period to be m. We now form the normalized scattergram of λ ordered pairs or, in the language of Marsaglia (Ref. 3), the 2-lattice of ordered pairs $(x_0/m, x_1/m)$, $(x_1/m, x_2/m)$, $(x_2/m, x_3/m)$, $(x_3/m, x_4/m)$, ..., $(x_{\lambda-1}/m, x_\lambda/m)$ where λ is the period of the generator. These points will lie inside a unit square of length one (Fig. 1). We shall now pass lines through the pairs. In Fig. 2 we have passed a family of vertical lines and a family of diagonal ones through a scattergram of rectangular points in the unit square. Of course, no linear congruential generator could produce a rectangular scattergram. We have chosen a rectangular one because it is a simple example. If the distance between the vertical lines is one unit, then that between the diagonal lines is $1/\sqrt{2}$ or approximately 0.7 unit. It is clear therefore that for this simple example, the distance between the diagonal lines is smaller than that between the vertical ones. In general, one can pass several families of lines through all pairs of points of the scattergram of a linear congruential generator. In Fig. 2, there are 100 points in the square, 10 vertical lines and 19 diagonal ones. Hence, the average number of points per line for the vertical family is 10 and the average number for the diagonal one is $100/19 \approx 5$. We note that the average number of points per line is smaller for the family whose lines are separated from each other by the smaller distance. All pairs that lie on the same line obviously

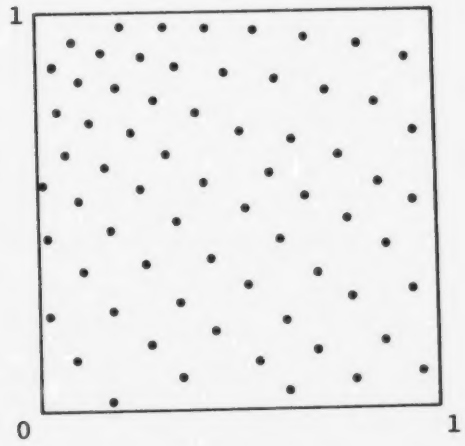

FIGURE 1 - Scattergram of normalized ordered pairs

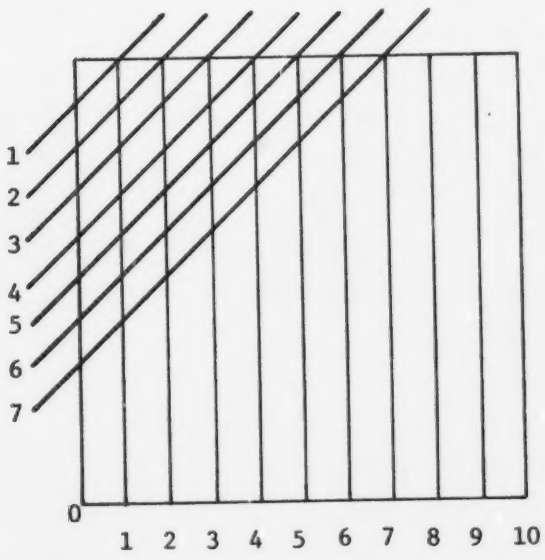

FIGURE 2 - Vertical and diagonal lines passing through a simple 2-lattice

satisfy the same linear relation. We seek, therefore, a linear congruential generator for which the distance between consecutive lines is small for every possible family of lines that pass through its scattergram of points. For such a generator, the number of pairs from the sequence linked by the same relation is small. Actually in the spectral test, we single out the family whose distance between lines is maximal in the set of all families whose lines pass through the pairs of the scattergram. This family can be thought of as the worst case family. The constants a and m are chosen so that the distance between consecutive lines of the worst case family is minimized. It turns out that the constant c does not affect the distribution of pairs of random numbers. The effect of c is to displace all pairs by the same fixed amount (Ref. 1, p 93). Constants a and m thus chosen are optimal for producing pairs of random numbers that are not linearly correlated by the same relation. A similar analysis applies to triples, quadruples, quintuples and sextuples of random numbers. In these cases, we are interested in calculating the distances between hyperplanes instead of those between lines. The random number generator that passes the spectral test is the one whose constants a and m have been chosen so as to minimize the distance between the hyperplanes in t-space (t = 2, ..., 6) of the worst case family of such hyperplanes containing the t-tuples $(x_1, x_2, \ldots, x_t)$, where each x_i comes from the linear congruential sequence. The reciprocal of this minimum distance is called by Knuth the t-dimensional accuracy of the generator and is denoted by ν_t.

4.0 THE SPECTRAL TEST COMPUTER PROGRAM

An efficient computational procedure for the spectral test is given in Ref. 1, pp 98-100. We have programmed this algorithm in Xerox extended FORTRAN for the Honeywell 560. The algorithm is iterative in the sense that the 2-dimensional accuracy is calculated from a type of Euclidean algorithm, and then the t-dimensional accuracy is calculated from the (t - 1) - dimensional accuracy through a process resembling the Gram-Schmidt orthogonalization for values of t from 3 to 6. When this orthogonalization comes to an end, an exhaustive search is made to see whether the value of ν_t can be further lowered. The value obtained by the exhaustive search is the final value output by the computer program as the t-dimensional accuracy of the random number generator.

The programming of the algorithm given in Knuth required extensive debugging. This was done by comparing the values for the 2-, 3-, 4-, 5- and 6- dimensional accuracies of linear congruential generators obtained by Ref. 1, pp 102-103 with those from our program. Now, the Honeywell 560 gives six significant digits of accuracy in single precision and 15 digits in double precision. Some of the moduli cited by Knuth are 11-digit numbers. Some places in the algorithm calculate integers of the size of m^2 i.e., a 22-digit number. It is clear, therefore, that the double precision of the Honeywell 560 does not allow an exact calculation of all intermediate variables of the spectral test. It was necessary to write multiple precision subroutines for addition, subtraction, multiplication, division, rounding off, raising to a power and finding square roots. The algorithms that we chose for addition, subtraction, multiplication and division are the standard pencil and paper methods. These are presented in detail in Ref. 1, pp 250-258. The multiple precision exponentiation routine is based on the multiple precision multiplication routine and on the binary decomposition of the exponent.

The square-root routine is a modified Newton-Raphson procedure that calculates the floor of the square root of a multiple precision integer. (The floor of a real number is the greatest integer smaller than it.)

The spectral test algorithm comprises 11 sections. Initially, we tried to replace each section by a block of FORTRAN where additions, subtractions, multiplications and divisions of the algorithm would be replaced by calls to multiple precision subroutines. We found however that some subsections of the algorithm were sufficiently complicated to justify two or three blocks of FORTRAN. In its final form the spectral test computer program comprises the main program and 29 subroutines of which three are input/output subroutines, 14 are multiple precision calculator routines and 12 are routines corresponding to the 11 sections of the spectral test. Table I gives a breakdown of these three types of subroutines.

Comment cards inserted in the program give a description of the function of each of the global variables used. Each subroutine is followed by a comment section where its purpose and the function of its parameters are described. All subroutines called by it and all those that call it are also listed.

TABLE I

Subroutines of the spectral test

INPUT/OUTPUT	INPUTS, PRINTS, TABLE
CALCULATOR MULTIPLE PRECISION	ADD, COPY, DIVIDE, ESTIMATEROOT, FASTDIV, FLOORNEG, INNERPROD, MULT, NEGDIFF, REMOVELDGZEROS, ROUND, SQROOT, SUBTRACT, SWAP.
ALGORITHM	INITIALIZE (S1) EUCLIDEAN (S2) TEST (S2 AND S3) FINDACCUR (PART 1 OF S3) STARTUVMATRICES (PART 2 OF S3) ENLARGEUV (S4) SETTROW (S4) TRANSFORM (S5) TEST2 (S4 AND S6) ASSIGNZ (S8) SEARCH (S8, S9, S10, S11) TEST3 (PART 2 OF S10)

5.0 HOW TO USE THE SPECTRAL TEST COMPUTER PROGRAM

According to Ref. 1, pp 101-102, it appears that if the t-dimensional accuracy ν_t exceeds $2^{30/t}$ for $2 \leq t \leq 6$, we will have found a good generator. For certain purposes, we would like a criterion that is independent of the modulus m, so that we can say that a particular multiplier a is good or bad with respect to the set of all other multipliers for the given m. Knuth has chosen as criterion

$$\mu_t = \pi^{t/2} \nu_t^t / (t/2)! m \quad 2 \leq t \leq 6 \qquad [7]$$

because this is the volume of an ellipsoid in t-space defined by the relation

$$(x_1 m - x_2 a - \ldots - x_t a^{t-1})^2 + x_2^2 + \ldots + x_t^2 \leq \nu_t^2 \qquad [8]$$

Now, in the formulation of the spectral test as a quadratic programming problem, ν_t is shown to be equal to

$$\min \{(mx_1 - ax_2 - a^2 x_3 - \ldots - a^{t-1} x_t)^2 + x_2^2 + x_3^2 + \ldots + x_t^2\}$$
$$(x_1, \ldots, x_t) \neq$$
$$(0, 0, 0, \ldots, 0) \qquad [9]$$

i.e. ν_t is the distance of the origin from the hyperplane defined by

$$x_1 m - x_2 a - \ldots - x_t a^{t-1} = x_1$$

The further this hyperplane is from the origin, the greater the volume of the ellipsoid [8] will be and vice versa. The formulae [7] take on the form

$$\mu_2 = \pi \nu_2^2 / m, \quad \mu_3 = (4/3) \pi \nu_3^3 / m, \quad \mu_4 = \pi^2 \nu_4^4 / 2m$$
$$\mu_5 = 8\pi^2 \nu_5^5 / 15m, \quad \mu_6 = \pi^3 \nu_6^6 / 6m \qquad [10]$$

Knuth says that a multiplier for a linear congruential generator passes the spectral test if $\mu_t > 0.1$ $(2 \leq t \leq 6)$ and it passes it very successfully if $\mu_t > 1$ $(2 \leq t \leq 6)$. A high value of μ_t $(2 \leq t \leq 6)$ means that we have an unusually good multiplier for the given m. In general, when searching for a good linear congruential generator, we choose as high a value of m as possible and then a value of a that gives us high values of μ_t $(2 \leq t \leq 6)$. The ultimate test, however, for a good generator is that the values of ν_t $(2 \leq t \leq 6)$ be large.

The FORTRAN code for the spectral test is in a file called SPECTRAL and the object code in one called SPECTRU (SPECTRAL RUN UNIT). To execute the spectral test in interactive mode it suffices to give the instruction

! S SPECTRU

The computer then asks for input. The user types in the base of the number system and then presses the carriage return key. The computer then asks for the number of digits in the multiplier. The user types in this value and then presses the carriage return key. The computer next asks for the number of digits in the modulus. The user supplies this value, presses the carriage return key and the computer will respond with the three values just input by the user. Suppose the user wishes to test the generator

$$x_n = (2718281829)\, x_{n-1} + 13745031166 \bmod 2^{35} \qquad [11]$$

The user inputs 10 for the base, 10 for the number of digits in the multiplier and 11 for the number of digits in the modulus since $2^{35} = 34359738368$. The user must press the carriage return key after each integer is typed in, otherwise the spectral test program will not function properly. After typing in the number of digits of the modulus, the computer responds with

IBASE = 10

NDIMMULT = 10

NDIMMOD = 11

i.e. the base of the number system is 10 and the multiplier and modulus are 10- and 11-digit numbers respectively. The user then inputs the 10 digits of the multiplier without separation by commas, presses the carriage return key and then inputs the 11 digits of the modulus without separation by commas and presses the carriage return key again. The computer responds with the message

MULTIPLIER IS 2718281829

MODULUS IS 34359738368

VALUES OF S
PRIOR TO (2nd column)
AND FOLLOWING (3rd column)
EXHAUSTIVE SEARCH

T	S	
2	22939188896	22939188896
3	2723830	2723830
4	148116	148116
5	10782	10782
6	2914	2914

The first two lines are self-explanatory. The integers in the second column under the variable S are the t-dimensional accuracies of the generator under consideration for t going from 2 to 6, i.e. 148116 is the 4-dimensional accuracy of the generator [11] before the exhaustive search is carried out, and it is also the 4-dimensional accuracy obtained at the end of S7 of the algorithm. The values given in column

three are those of the t-dimensional accuracy after the exhaustive search (steps S8-S11 of algorithm S of Ref. 1) has been carried out. The values of the accuracy before and after the exhaustive search were calculated because we were looking for a linear congruential generator where the exhaustive search reduced the t-dimensional accuracy further. Knuth (Ref. 1, p 99) says that he has been unable to find such an example. We therefore tested many examples in the hope of finding one where the exhaustive search did something.

After obtaining the t-dimensional accuracies of a linear congruential generator, the user may execute the merit calculator subroutine by giving the instruction S MERITRU. The computer prompts for more information. The user supplies the t-dimensional accuracies ($2 \leq t \leq 6$) obtained from the spectral test program and presses the carriage return key again. The computer responds with output of the form shown below

```
S MERITRU
?34359738368
?22939188896
?2723830
?148116
?10782
?2914
MERIT OF THE CHOSEN MULTIPLIER
FOR THE INPUTTED MODULUS
I    MERIT
2     2.10
3      .55
4     3.15
5     1.85
6     3.72
```

6.0 ASSESSMENT OF COMMONLY USED GENERATORS AND PROPOSAL OF ALTERNATIVES

Our spectral test program was tried on the random number generators 1-27, 29 and 30 of page 102 of Ref. 1. We obtained the same results as Knuth except for the generators on lines 8, 11 and 22, and even then the differences were slight. For the generator of line 22 whose inputs are $a = 2^{16} + 3$, $m = 2^{29}$, i.e. for RANDU, our spectral test program gave us $\nu_2^2 = 536936458$ while Knuth has $\nu_2^2 = 536805386$. In line 11 of Table I, Knuth has obtained $\nu_4^2 = 146116$ for the random number generator whose inputs to the spectral test are:

$$a = 2718281829,\ m = 2^{35}$$

We have obtained $\nu_4^2 = 148116$ from our computer program. Furthermore when we computed μ_4 from formula 37 of page 102 of Ref. 1, i.e.

$$\mu_4 = \pi^2 \nu_4^4 / 2m$$

we obtained $\mu_4 = 3.15$ for $\nu_4^4 = 148116$ and $\mu_4 = 3.07$ for $\nu_4^2 = 146116$. Now, the value of μ_4 shown in line 11 of Table I on page 103 of Ref. 1 is precisely 3.15, i.e. the value obained from 148116. We think therefore that the value ν_4^2 of line 11 of Table I should be 148116 rather than 146116. In line 8 of Table I, Knuth has obtained $\mu_2 = 3.37$ by using the formula

$$\mu_2 = \pi \nu_2^2 / m$$

given as eq. 37 of page 102 of Ref. 1. In this case, the calculation of μ_2 turns out to be simple because $m = 10^{10}$. We obtain

$$\mu_2 = (3.1415926536)\ (10721093248)/10^{10}$$
$$= 33681307786 \cdot 4773628928 * 10^{-10}$$
$$= 3.36813077864773628928$$
$$= 3.37$$

In addition to the examples of Knuth's book, we tested two IMSL routines called GGUBS and GGUBT. GGUBS is a linear congruential generator of the form

$$x_n = (16807)\ x_{n-1} \bmod 2147483647 = 2^{31} - 1$$

$$x_i \text{ is an integer }, 1 < x_i < 2147483647$$

while GGUBT is a linear congruential generator of the form

$$x_n = (397204094)\ x_{n-1} \bmod 2147483647$$

$$x_i \text{ is an integer, } 1 < x_i < 2147483647$$

The results we have obtained from the spectral test for these generators are presented in Table II and graphed in Fig. 3.

We used this test to find three linear congruential generators which had very high values of accuracy and merit. These were

$$x_n = (2050207317)\ x_{n-1} + 3 \bmod 2147483648 = 2^{31} \qquad [12]$$

$$x_0 = \text{any 9-digit number even or odd}$$

We divide x_n by $2^{31} - 1$ to obtain a uniform deviate between 0 and 1. The second generator that we propose is

TABLE II

Accuracies and merits of six random number generators

	Accuracies					Merits				
	ν_2	ν_3	ν_4	ν_5	ν_6	μ_2	μ_3	μ_4	μ_5	μ_6
RANDU	536936458	118	116	116	116	3.14	ϵ^5	ϵ^4	ϵ^3	0.02
GGUBS	282475250	408197	21682	4439	895	0.41	0.51	1.08	3.22	1.73
GGUBT	767608202	692941	29187	4829	760	1.12	1.13	1.96	3.97	1.06
[12]	1930254976	969058	49562	5274	862	2.82	1.86	5.64	4.95	1.54
[13]	691573378	986334	25058	5304	780	1.01	1.91	1.44	5.02	1.14
[14]	413797280	558970	12570	1690	498	2.42	3.26	1.45	1.15	1.19

Note: $\epsilon = .1$

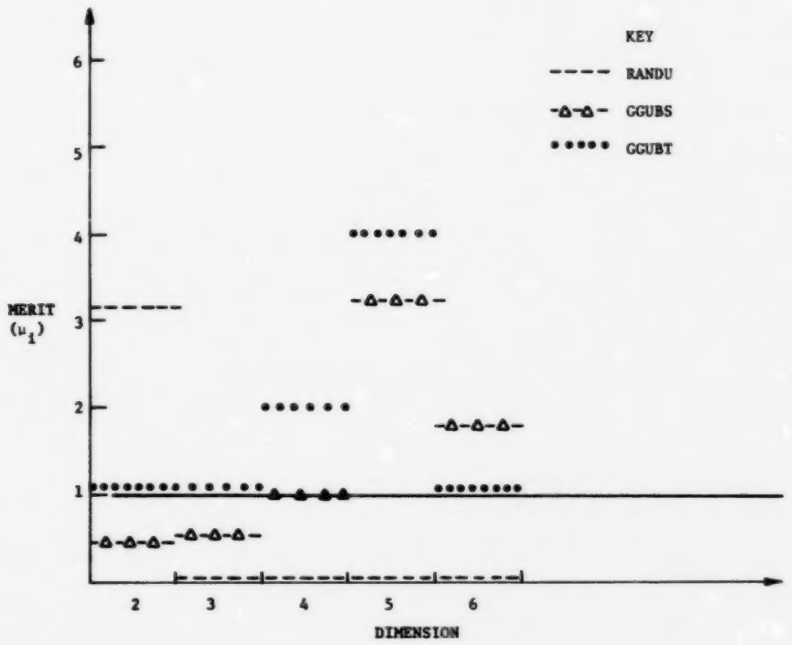

FIGURE 3 - Graph of merits of existing random number generators

$$x_n = (2016296033)\ x_{n-1} + 1824726041 \bmod 2^{31} \qquad [13]$$

x_0 = any 9-digit number even or odd

The third generator that we propose is

$$x_n = (464680355)\ x_{n-1} \bmod 2^{31} \qquad [14]$$

x_0 = any 9-digit odd number

We note that the multiplier of the generator defined by [12] $2050207317 = 4(2^{28} + 244116372) + 1$ and its constant 3 is relatively prime to 2^{31}. Hence the period of this generator is 2^{31} by Theorem 1-1. Similarly the multiplier of [13], namely 2016296033, is equal to $4(2^{28} + 235638552) + 1$, the constant of [13], namely 1824726041, has no factor of 2 and hence [13] has a period of 2^{31} by Theorem 1-1. The

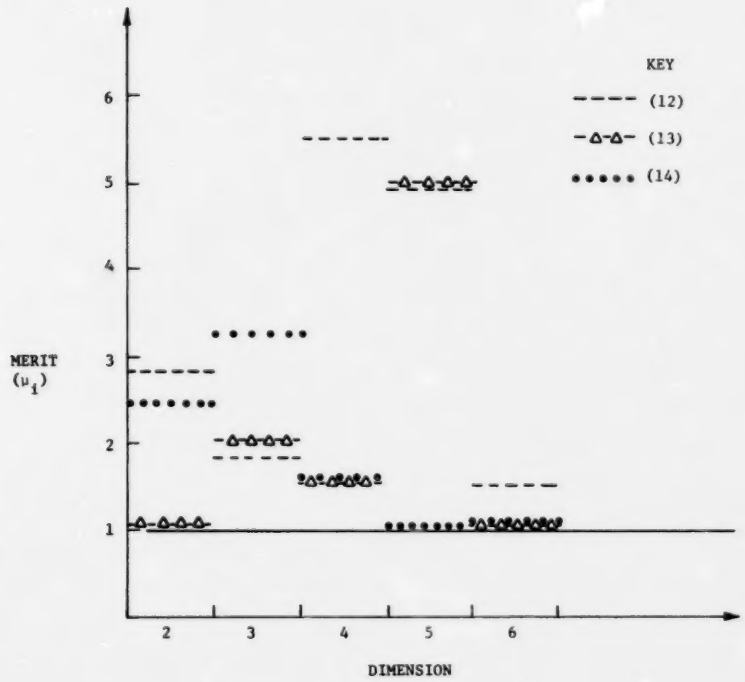

FIGURE 4 - Graph of merits of proposed random number generators

multiplier of [14], namely 464680355, is equal to $2^{28} + 8(24530612) + 3$ and hence is a primitive element modulo 2^{31} by (a) of Theorem 1-3. The two conditions of Theorem 1-2 are now satisfied which means that [14] has a period of length $\lambda(2^{31}) = 2^{29}$. The inputs to the spectral test for generators [12], [13] and [14] will be respectively $a = 2050207317$, $m = 2^{31}$, $a = 2016296033$, $m = 2^{31}$, $a = 464680355$, $m = 2^{29}$. The accuracies and merits of these three generators are given in Table II and graphed in Fig. 4.

As one can see from Fig.3, RANDU fails the spectral test completely, GGUBS passes it and GGUBT passes it very successfully. All three generators that we propose pass the spectral test with complete success as can be seen from Fig. 4.

The computer listings of the generators [12], [13] and [14] are found in Appendix A. The FORTRAN programs given in the appendix are machine-dependent in the sense that they will probably not work on computers other than the Honeywell 560.

7.0 NORMAL DEVIATES

As was pointed out in the introduction, random phenomena are simulated, using the Monte Carlo method, by random sampling from uniform and normal distributions. We have dealt with uniform deviates in Chapter 6.0 so now we consider normal deviates. Normal deviates or, for that matter, deviates from other probability distributions are generated from uniform deviates with the aid of a suitable conversion function. The algorithm we have chosen for normal deviates is the polar method developed by G.E.P. Box, M.E. Muller, and G. Marsaglia (see Annals Math. Stat. Vol. 28, p 610, 1958). It is also to be found in Ref. 1, p 117. We have used generators [12] and [14] of Chapter 6.0

as generators of uniform deviates for the normal deviate program. The computer listing of this program is to be found in Appendix A.

8.0 CONCLUSION

The results of this work show that consecutive triples, quadruples, quintuples and sextuples generated by RANDU fall into relatively few hyperplanes. This means that there exists a small number of linear equations in three variables such that any three consecutive random numbers generated by RANDU satisfy one of these equations. However, the spectral test shows that the 2-dimensional accuracy of RANDU is excellent. The authors think therefore that RANDU could profitably be used in a simulation where each unit requires one or two random numbers to produce a result or generate an event. We think that in this case simulation of the random factor of the unit would be sufficiently random. However, if the unit in the simulation requires three consecutive draws from RANDU to produce a result or generate an event, the sample mean estimate of this result calculated from a large number of replications may differ from the same estimate calculated from the same number of replications with a random number generator of high 3-dimensional accuracy. In other words, use of random number generators with a low 3-dimensional accuracy could give biased estimates. The last statement seems plausible to the authors but has not been vigorously validated. In any case, if the reader used the IMSL routine GGUBT or any one of the proposed generators [12], [13] or [14], he would avoid these problems and obtain reasonable random numbers. In general, the accuracy (ν_t, $2 < t < 6$) of a random number generator increases as a function of the modulus and of the multiplier. The reader should there fore always aim for generators with large moduli and large multipliers. The latter assertion is heuristically plausible since numbers produced by a generator with a small multiplier would tend to be equally spaced and those produced by a generator with a small modulus would cycle quickly.

9.0 ACKNOWLEDGEMENTS

The authors would like to thank Dr B. Evans for telling them about Knuth's "The Art of Computer Programming", Volume 2, which is an excellent reference on random number generators. This publication was very useful to them in the course of their work.

10.0 REFERENCES

1. Knuth, D.E., "The Art of Computer Programming, Seminumerical Algorithms", Vol. 2, Addison Wesley, Reading, 2nd Edition, 1981.

2. Coveyou, R.R. and MacPherson, R.D., "Fourier Analysis of Uniform Random Number Generators", JACM 14, pp 100-119, 1967.

3. Marsaglia, G., "Regularities in Congruential Random Number Generators", Numerische Mathematik, Vol. 16, pp 8-10, 1970.

4. Marsaglia, G. and Bray, T.A., "One-line Random Number Generators and their Use in Combinations", CACM, Vol. 2, pp 757-759, 1968.

APPENDIX A

Listing of Three Generators of Uniform Deviates and One Generator of Normal Deviates for the Honeywell 560

```
 1.000 C      SUBROUTINE UNIFDEV1 (UNIF)
 2.000 C         THIS IS A RANDOM NUMBER GENERATOR WITH:
 3.000 C              A = 2050207317, C = 3, M = 2147483648
 4.000 C      EACH  NEW RANDOM NUMBER GENERATED IS WRITTEN
 5.000 C      OVER THE PREVIOUS ONE IN THE FILE CONNECTED TO LOGICAL
 6.000 C      UNIT 6. UNIF IS THE NORMALISED RANDOM RESULT, A REAL
 7.000 C      NUMBER BETWEEN 0 AND 1.
 8.000 C
 9.000        DEFINE FILE 6 (1,4,E,IK)
10.000 X      DO 10 J=1,10
11.000        READ (6'1,89) IAL
12.000        CALL RAND1 (IAL,UNIF)
13.000        WRITE (6'1,89) IAL
14.000 X      OUTPUT UNIF
15.000 X10    CONTINUE
16.000   89   FORMAT (I)
17.000        STOP
18.000        END
19.000 C
20.000        SUBROUTINE RAND1 (IAL,UNIF)
21.000        DOUBLE PRECISION MODLESS1
22.000        DATA MODLESS1/2147483647/
23.000        IAL = IAL * 2050207317 + 3
24.000        IF (IAL .GE. 0) GO TO 10
25.000           IAL = IAL + 2147483647 + 1
26.000   10   UNIF = IAL/MODLESS1
27.000        RETURN
28.000        END
```

```
 1.000 C      SUBROUTINE UNIFDEV2 (UNIF)
 2.000 C         THIS IS A RANDOM NUMBER GENERATOR WITH:
 3.000 C              A = 2016296033, C = 1824726041, M = 2147483648
 4.000 C      EACH NEW RANDOM NUMBER GENERATED IS WRITTEN OVER THE
 5.000 C      PREVIOUS ONE IN THE FILE CONNECTED TO LOGICAL UNIT 7.
 6.000 C      UNIF IS THE NORMALIZED RANDOM RESULT, A REAL NUMBER
 7.000 C      BETWEEN 0 AND 1.
 8.000 C
 9.000        DEFINE FILE 7 (1,4,E,IK)
10.000 X      DO 10 J=1,100
11.000        READ (7'1,89) IAL
12.000        CALL RAND2 (IAL,UNIF)
13.000        WRITE (7'1,89) IAL
14.000 X      OUTPUT UNIF
15.000 X10    CONTINUE
16.000   89   FORMAT (I)
17.000        STOP
18.000        END
19.000 C
20.000        SUBROUTINE RAND2 (IAL,UNIF)
21.000        DOUBLE PRECISION MODLESS1
22.000        DATA MODLESS1/2147483647/
23.000        IAL = IAL * 2016296033 + 1824726041
24.000        IF (IAL .GE. 0) GO TO 10
25.000           IAL = IAL + 2147483647 + 1
26.000   10   UNIF = IAL/MODLESS1
27.000        RETURN
28.000        END
```

```
 1.000 C      SUBROUTINE UNIFDEV3 (UNIF)
 2.000 C         THIS IS A RANDOM NUMBER GENERATOR WITH:
 3.000 C              A = 464680355,  C = 0,  M = 2147483648
 4.000 C      EACH  NEW RANDOM NUMBER GENERATED IS WRITTEN OVER THE
 5.000 C      PREVIOUS ONE IN THE FILE CONNECTED TO LOGICAL UNIT 8.
 6.000 C      UNIF IS THE NORMALIZED RANDOM RESULT, A REAL NUMBER
 7.000 C      BETWEEN 0 AND 1.
 8.000 C
 9.000        DEFINE FILE 8 (1,4,E,IK)
10.000 X      DO 10 J=1,100
11.000        READ (8'1,89) IAL
12.000        CALL RAND3 (IAL,UNIF)
13.000        WRITE (8'1,89) IAL
14.000 X      OUTPUT UNIF
15.000 X10    CONTINUE
16.000   89   FORMAT (I)
17.000        STOP
18.000        END
19.000 C
20.000        SUBROUTINE RAND3 (IAL,UNIF)
21.000        DOUBLE PRECISION MODLESS1
22.000        DATA MODLESS1/2147483647/
23.000        IAL = IAL * 464680355
24.000        IF (IAL .GE. 0) GO TO 10
25.000           IAL = IAL + 2147483647 + 1
26.000   10   UNIF = IAL/MODLESS1
27.000        RETURN
28.000        END
```

```
 1.000 C      SUBROUTINE  NORMDEV (X1,X2)
 2.000 C          THIS ROUTINE IS A PROGRAMMED VERSION OF THE
 3.000 C      POLAR  METHOD FOR OBTAINING A NORMAL DISTRIBUTION.
 4.000 C      IT  USES TWO RANDOM NUMBER GENERATORS TO OBTAIN TWO
 5.000 C      INDEPENDENT  NORMALLY DISTRIBUTED VARIABLES.
 6.000 C
 7.000        GLOBAL MODLESS1
 8.000        DOUBLE PRECISION MODLESS1
 9.000        DATA MODLESS1/2147483647/
 9.200 X      DEFINE FILE 18 (100,8,E,IK)
 9.500 X      DO 10 J = 1,100
10.000        S = 1
11.000        REPEAT 30,WHILE (S .GE. 1)
12.000           CALL UNIFDEV1 (UNIF1)
13.000           CALL UNIFDEV3 (UNIF2)
14.000           V1 = 2 * UNIF1 - 1
15.000           V2 = 2 * UNIF2 - 1
16.000           S = V1**2 + V2**2
17.000   30   CONTINUE
18.000        VALUE = SQRT (-2 * LOG(S)/S)
19.000        X1 = V1 * VALUE
20.000        X2 = V2 * VALUE
20.500 X      WRITE (18'J,1000) X1,X2
20.600 X10    CONTINUE
20.700 X1000  FORMAT (2F10.6)
21.000        STOP
22.000        END
```

CRDV R-4319/84 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. G0A 1R0

"Un codage en FORTRAN de l'essai spectral pour évaluer la répartition et le mérite des générateurs congruents linéaires"
par R. Carling et K. Levine

Dans ce rapport on présente un algorithme mathématique, connu sous le nom d'essai spectral, comme moyen d'évaluer la répartition et le mérite des générateurs congruents linéaires. On explique le mode d'emploi de notre codage en FORTRAN de l'essai spectral pour ordinateur Honeywell 560. On évalue le mérite de générateurs souvent employés en simulation tels que RANDU (NUMGEN) et de deux générateurs de IMSL. On propose d'autres générateurs pour remplacer ceux qui échouent l'essai spectral.

CRDV R-4319/84 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. G0A 1R0

"Un codage en FORTRAN de l'essai spectral pour évaluer la répartition et le mérite des générateurs congruents linéaires"
par R. Carling et K. Levine

Dans ce rapport on présente un algorithme mathématique, connu sous le nom d'essai spectral, comme moyen d'évaluer la répartition et le mérite des générateurs congruents linéaires. On explique le mode d'emploi de notre codage en FORTRAN de l'essai spectral pour ordinateur Honeywell 560. On évalue le mérite de générateurs souvent employés en simulation tels que RANDU (NUMGEN) et de deux générateurs de IMSL. On propose d'autres générateurs pour remplacer ceux qui échouent l'essai spectral.

CRDV R-4319/84 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. G0A 1R0

"Un codage en FORTRAN de l'essai spectral pour évaluer la répartition et le mérite des générateurs congruents linéaires"
par R. Carling et K. Levine

Dans ce rapport on présente un algorithme mathématique, connu sous le nom d'essai spectral, comme moyen d'évaluer la répartition et le mérite des générateurs congruents linéaires. On explique le mode d'emploi de notre codage en FORTRAN de l'essai spectral pour ordinateur Honeywell 560. On évalue le mérite de générateurs souvent employés en simulation tels que RANDU (NUMGEN) et de deux générateurs de IMSL. On propose d'autres générateurs pour remplacer ceux qui échouent l'essai spectral.

CRDV R-4319/84 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 8800, Courcelette, Qué. G0A 1R0

"Un codage en FORTRAN de l'essai spectral pour évaluer la répartition et le mérite des générateurs congruents linéaires"
par R. Carling et K. Levine

Dans ce rapport on présente un algorithme mathématique, connu sous le nom d'essai spectral, comme moyen d'évaluer la répartition et le mérite des générateurs congruents linéaires. On explique le mode d'emploi de notre codage en FORTRAN de l'essai spectral pour ordinateur Honeywell 560. On évalue le mérite de générateurs souvent employés en simulation tels que RANDU (NUMGEN) et de deux générateurs de IMSL. On propose d'autres générateurs pour remplacer ceux qui échouent l'essai spectral.

DREV R-4319/84 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"A FORTRAN Coding of the Spectral Test for Assessing the Randomness and the Merit of Linear Congruential Generators"
by R. Carling and K. Levine

In this report, a mathematical algorithm known as the spectral test for assessing the randomness and merit of linear congruential generators is presented. Details of how to use our FORTRAN coding of the spectral test on the Honeywell 560 are given. Widely used generators such as RANDU (NUMGEN) and two IMSL generators are assessed by it. Alternative generators are proposed for bad ones not passing the spectral test.

DREV R-4319/84 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"A FORTRAN Coding of the Spectral Test for Assessing the Randomness and the Merit of Linear Congruential Generators"
by R. Carling and K. Levine

In this report, a mathematical algorithm known as the spectral test for assessing the randomness and merit of linear congruential generators is presented. Details of how to use our FORTRAN coding of the spectral test on the Honeywell 560 are given. Widely used generators such as RANDU (NUMGEN) and two IMSL generators are assessed by it. Alternative generators are proposed for bad ones not passing the spectral test.

DREV R-4319/84 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"A FORTRAN Coding of the Spectral Test for Assessing the Randomness and the Merit of Linear Congruential Generators"
by R. Carling and K. Levine

In this report, a mathematical algorithm known as the spectral test for assessing the randomness and merit of linear congruential generators is presented. Details of how to use our FORTRAN coding of the spectral test on the Honeywell 560 are given. Widely used generators such as RANDU (NUMGEN) and two IMSL generators are assessed by it. Alternative generators are proposed for bad ones not passing the spectral test.

DREV R-4319/84 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 8800, Courcelette, Que. G0A 1R0

"A FORTRAN Coding of the Spectral Test for Assessing the Randomness and the Merit of Linear Congruential Generators"
by R. Carling and K. Levine

In this report, a mathematical algorithm known as the spectral test for assessing the randomness and merit of linear congruential generators is presented. Details of how to use our FORTRAN coding of the spectral test on the Honeywell 560 are given. Widely used generators such as RANDU (NUMGEN) and two IMSL generators are assessed by it. Alternative generators are proposed for bad ones not passing the spectral test.

POOR

QUALITY

ORIGINAL